

# 1 Семейство ОС Unix/Linux

## 1.1 Напоминания

**Операционная система (ОС)** – занимает центральную часть в **системном ПО**, обеспечивая согласованное функционирование программных и аппаратных средств вычислительной системы. В ОС концентрируется большинство системных функций. Кроме ОС, к системному ПО относятся также **драйверы** и **системные утилиты**.

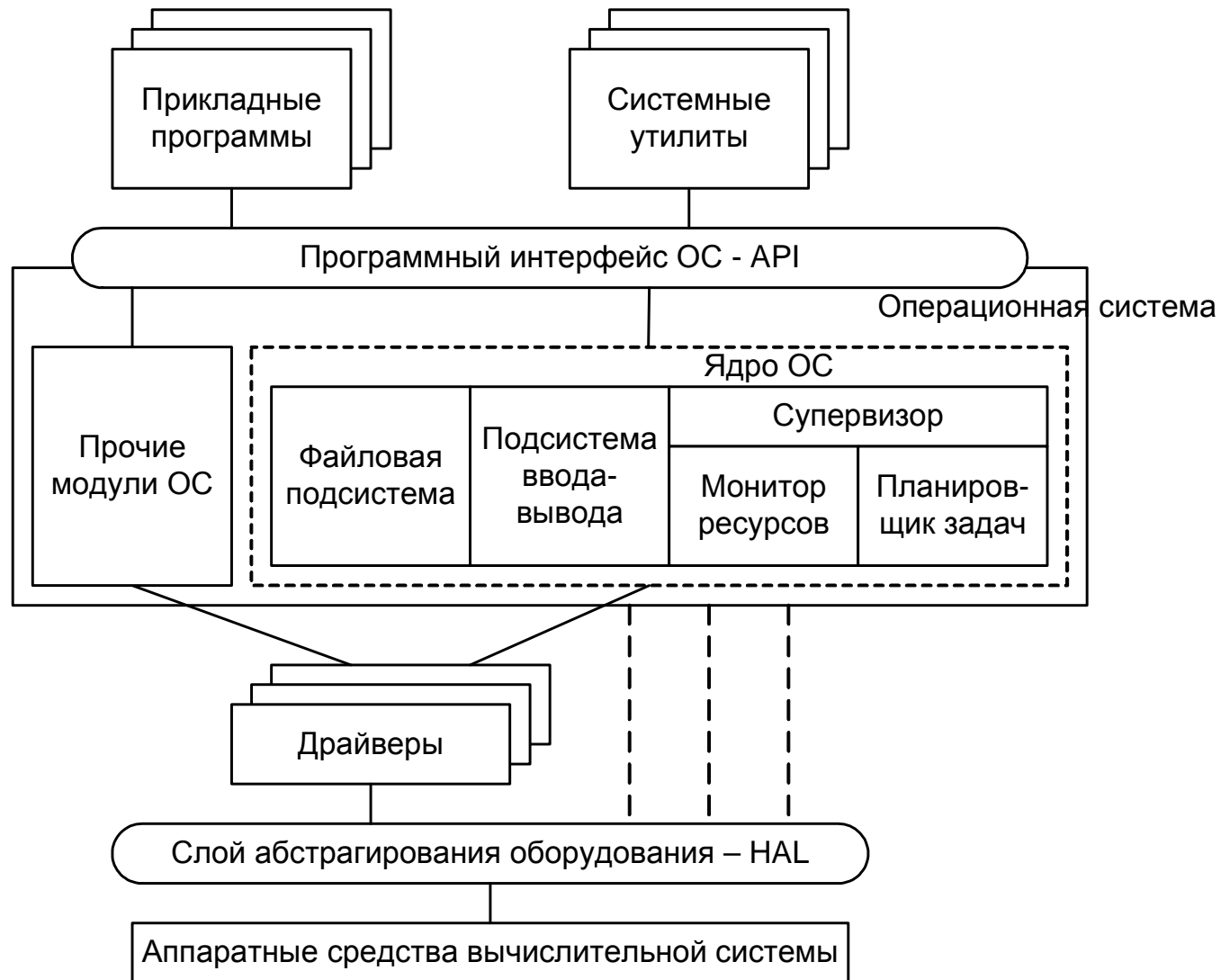
Операционная система:

- расширение машины, унифицированная надстройка, **виртуальная машина**
- менеджер ресурсов вычислительной системы

В ОС концентрируется большинство системных функций:

- управление процессами;
- управление ресурсами;
- управление вводом-выводом (в том числе файлами и файловой системой);
- обеспечение интерфейса с пользователем;
- общее управление и синхронизация.

## Операционные системы и среды: Семейство ОС Unix/Linux



Упрощенная структура операционной системы

***API – Application Programming Interface*** – интерфейс прикладного программирования, фактически программный интерфейс системы, подсистемы, компонента.

***HAL – Hardware Abstraction Layer*** – слой абстрагирования оборудования

***Kernel*** – ядро операционной системы, содержит основные ее подсистемы:

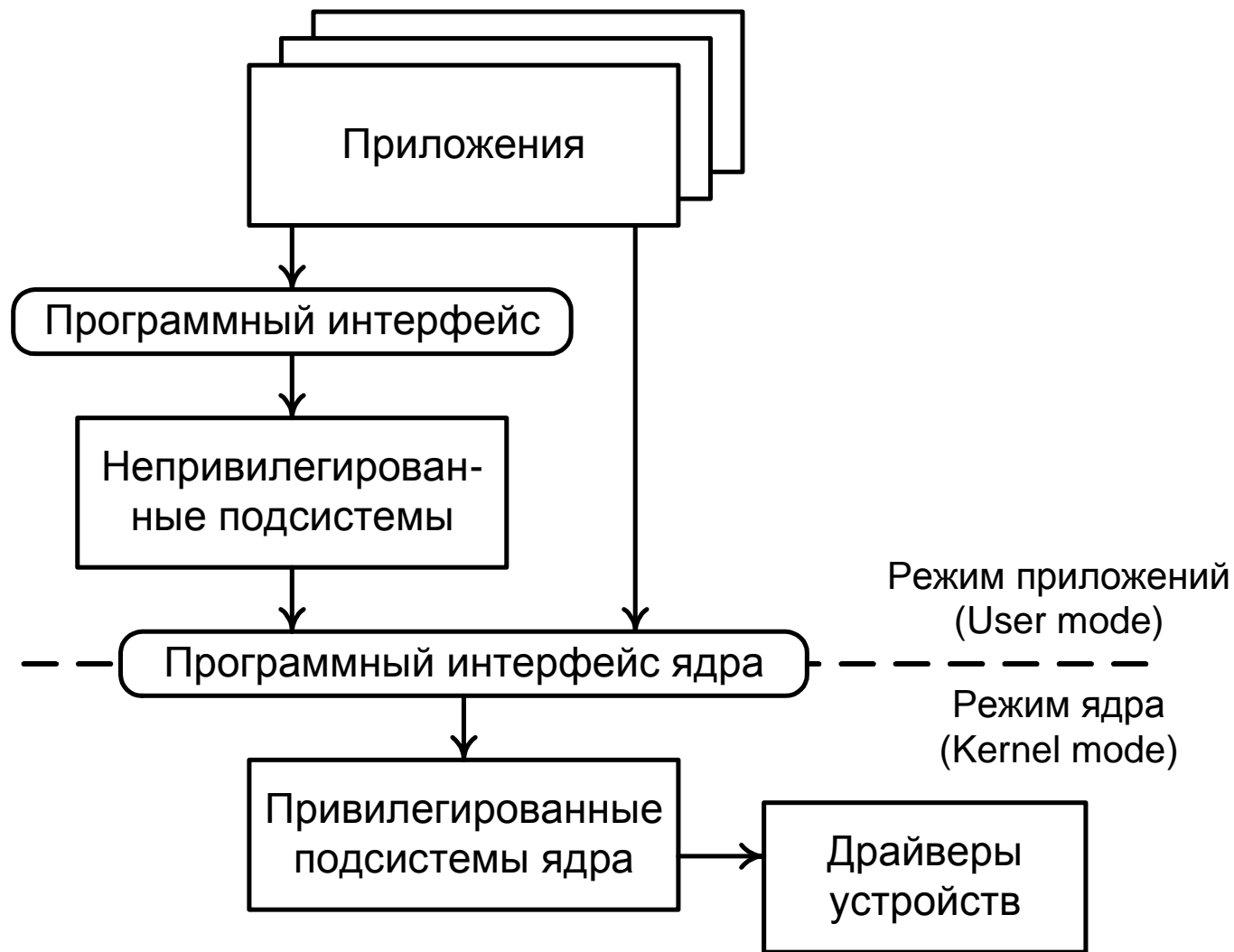
- подсистема файлов
- подсистема ввода-вывода
- монитор ресурсов (память и др.) – распределение ресурсов вычислительной системы процессам-пользователям
- планировщик (диспетчер) задач – распределение процессорного времени

В зависимости от архитектуры ОС, могут присутствовать также подсистемы вне ядра (непривилегированные).

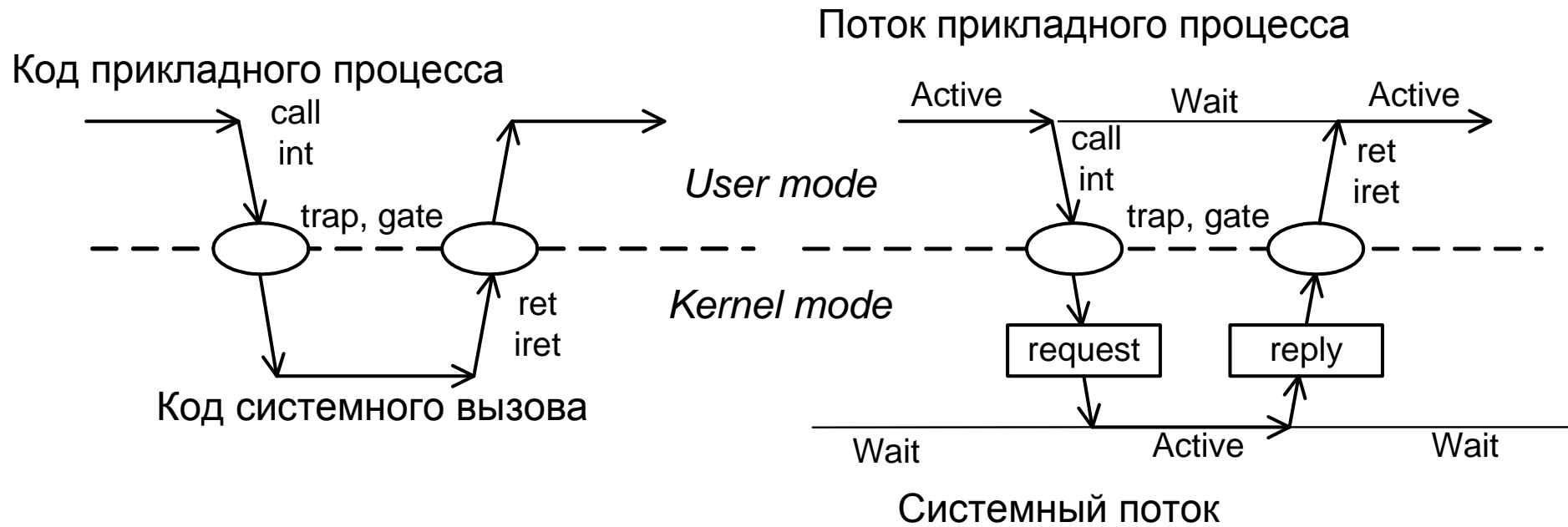
**Одноуровневые** ОС – код всех программ всегда выполняется на одном и том же уровне привилегий (простые однозадачные или специализированные системы)

**Двухуровневые** ОС – выделяются 2 уровня (режима) выполнения:

- Режим **приложения** (*User Mode*) – ограниченный в правах
- Режим **ядра** (*Kernel Mode*) – привилегированный



Уровни выполнения программ



а) переключение контекстов

б) переключение потоков

Переключение уровня выполнения кода

Для «классических» Unix-систем, а также для Linux характерна архитектура **монолитного** ядра (точнее, модульного **макро-ядра**). Некоторые ветви, например BSD UNIX – **микроядро**. Одна из наиболее известных UNIX-подобных (**Unix-like**) систем – ОС реального времени QNX – также микроядро.



**Вычислительный процесс (*Process*), образ, контекст, состояния** процесса.

**Вычислительный/программный поток (*Thread*), контекст, состояния** потока

**Виртуальная память: страницы, сегменты, отображение** страниц

**Файловая система: файлы, каталоги/директории, специальные типы файлов. Файловый ввод-вывод.**

**Событийное управление: прерывания (*interrupt*), исключения (*exception*), сигналы (*signal*), события (*event*), сообщения (*message*). *IPC* и *ISO*.**

.

## **1.2 Общая характеристика семейства ОС \*nix – Unix, Linux и Unix-подобных систем**

### **Характерные особенности:**

- Многозадачные (на основе разделения времени) и многопользовательские
- Переносимость на многие платформы благодаря исходному коду на С (в значительной мере, но не только)
- Единая иерархическая файловая система, унификация доступа к ресурсам посредством файловой системы
- Разнообразие поддерживаемых служб и другого ПО (благодаря переносимости ПО)
- Командная строка как основной интерфейс пользователя

## Операционные системы и среды: Семейство ОС Unix/Linux

- Единство и преемственность набора команд и программных интерфейсов (API) системы
- Единые концепции и идеология системы
- Следование единым стандартам

Некоторые современные ветви, семейства и подсемейства Unix-систем:

AIX (IBM), IRIX (Silicon Graphics), HP UX (Hewlett-Packard), Digital Unix (DEC-Compaq), Solaris (Sun Microsystems), Sun OS (они же), SCO Unix (Caldera Open Unix), BSD/NetBSD/FreeBSD/BSDI, Linux.

## **Краткая хронология:**

1961 – начало разработки многозадачной ОС Multics (AT&T, Bell Labs, MIT)

1969 – первая 16-разрядная версия Unix (Bell Labs).  
Практического значения не имела

1971 – «первая редакция», перенос на PDP-11 (заказ патентного отдела Bell Labs). Написана на ассемблере, содержала интерпретируемый язык B

1973 – «третья» и «четвертая редакции», ядро переписано на языке C. Первое официальное представление Unix.

1975 – «шестая редакция», впервые доступна вне Bell Labs. Полностью переписана на C.

1978 – первая версия BSD Unix (Berkley Software Distribution), на основе «шестой редакции»

## Операционные системы и среды: Семейство ОС Unix/Linux

- 1979 – «седьмая редакция», включает Bourne Shell и компилятор Kernighan&Ritchie C. Разработка переходит к Unix System Group. Выпуск 3BSD (на основе «седьмой редакции»)
- 1982 – System III (AT&T) как объединение известных версий
- 1983 – 4.2BSD, содержит реализацию TCP/IP. Выпуск System V. BSD и SV (а также их сочетание) – основа для большинства современных Unix-систем
- 1989 – System V Release 4, очередное объединение версий (System V, Sun OS, BSD)
- 90-е – разработка Linux (изначально для Intel 80386)

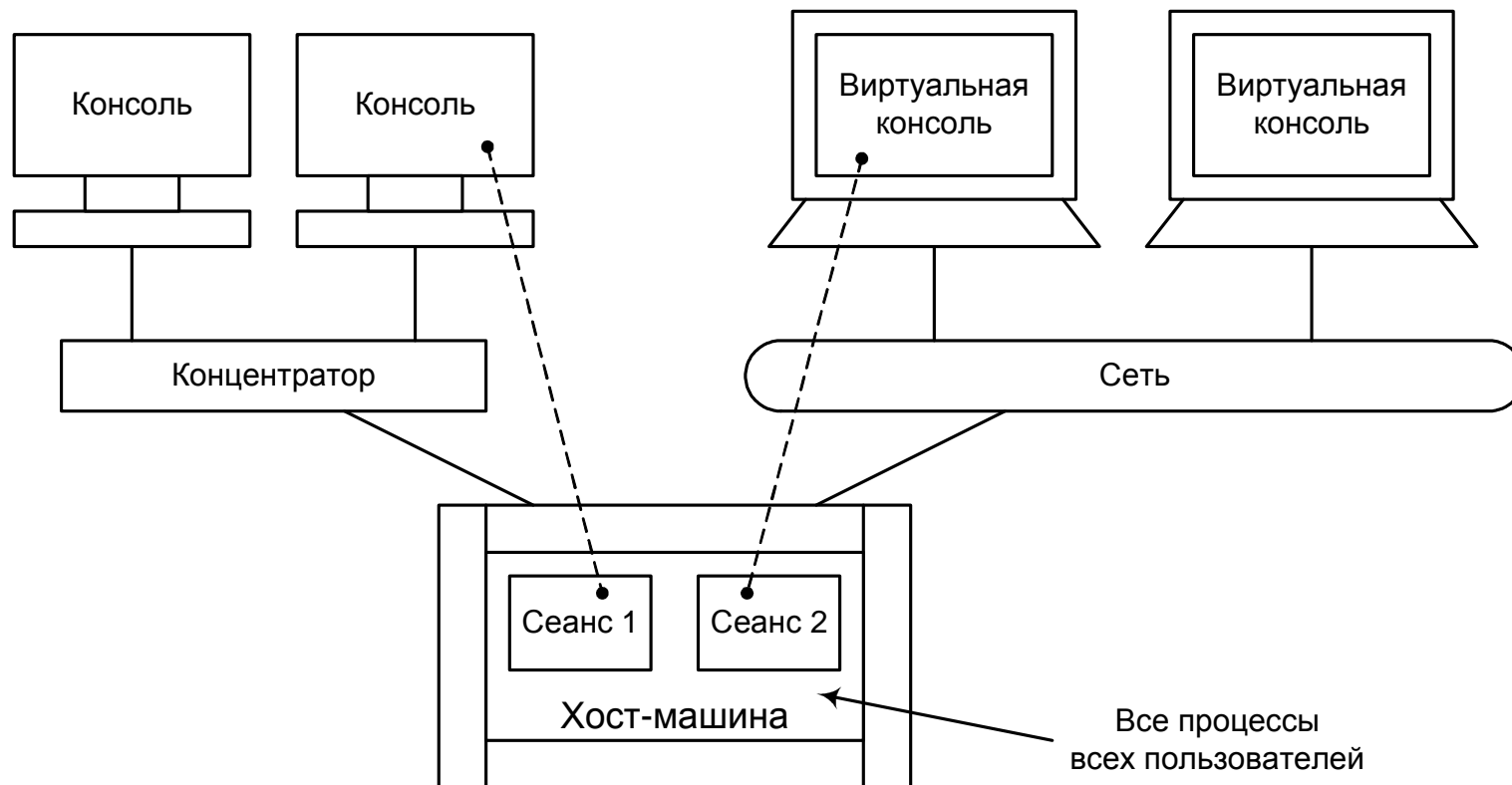
## **Некоторые стандарты:**

- POSIX (Portable Operating System Interface)
  - SUSv3 (Opengroup Single Unix Specification Version 3)
  - SVID (System V Interface Definition)
  - X/Open: CAE (Common Application Environment), XPG (X/Open Portability Guide)
  - ANSI C
- и др.

Организация, ответственная за выработку единых стандартов для Unix-систем и обладатель «торговой марки» UNIX – OpenGroup ([www.opengroup.com](http://www.opengroup.com)).

## 1.3 Структура Unix-систем

По историческим причинам Unix развивался как полноценно многопользовательская система.



Типичная конфигурация Unix-системы

Используемая первоначально техническая платформа – «средние» и мини-ЭВМ – предполагала множество пользователей, работающих с системой одновременно посредством **терминалов**, соединенных с центральной (**хост**) машиной различными каналами связи. Сейчас типичным терминалом стал обычный компьютер, а каналом связи – сеть.

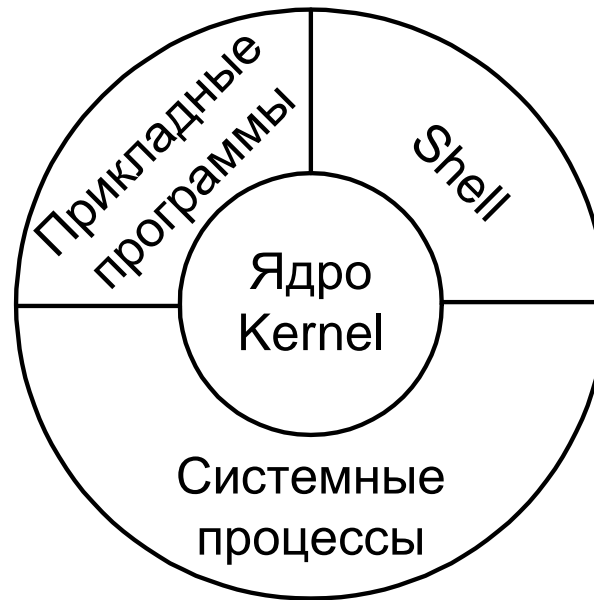
Частный случай – **виртуальная консоль** «автономной» Linux-системы без удаленного доступа. Одной из виртуальных консолей может быть графический интерфейс («рабочий стол»).

**Сеанс (session)** – все процессы пользователя, выполняемые начиная с момента входа в систему (login) и до выхода из нее (logout), связанные с одной консолью.

Одновременно может существовать несколько сеансов одного пользователя. Могут существовать процессы вне любого из пользовательских сеансов.



Структуру системы часто представляют упрощенно в виде ядра и взаимодействующих с ним процессов (типичная двухуровневая система):



Упрощенная структура Unix-системы

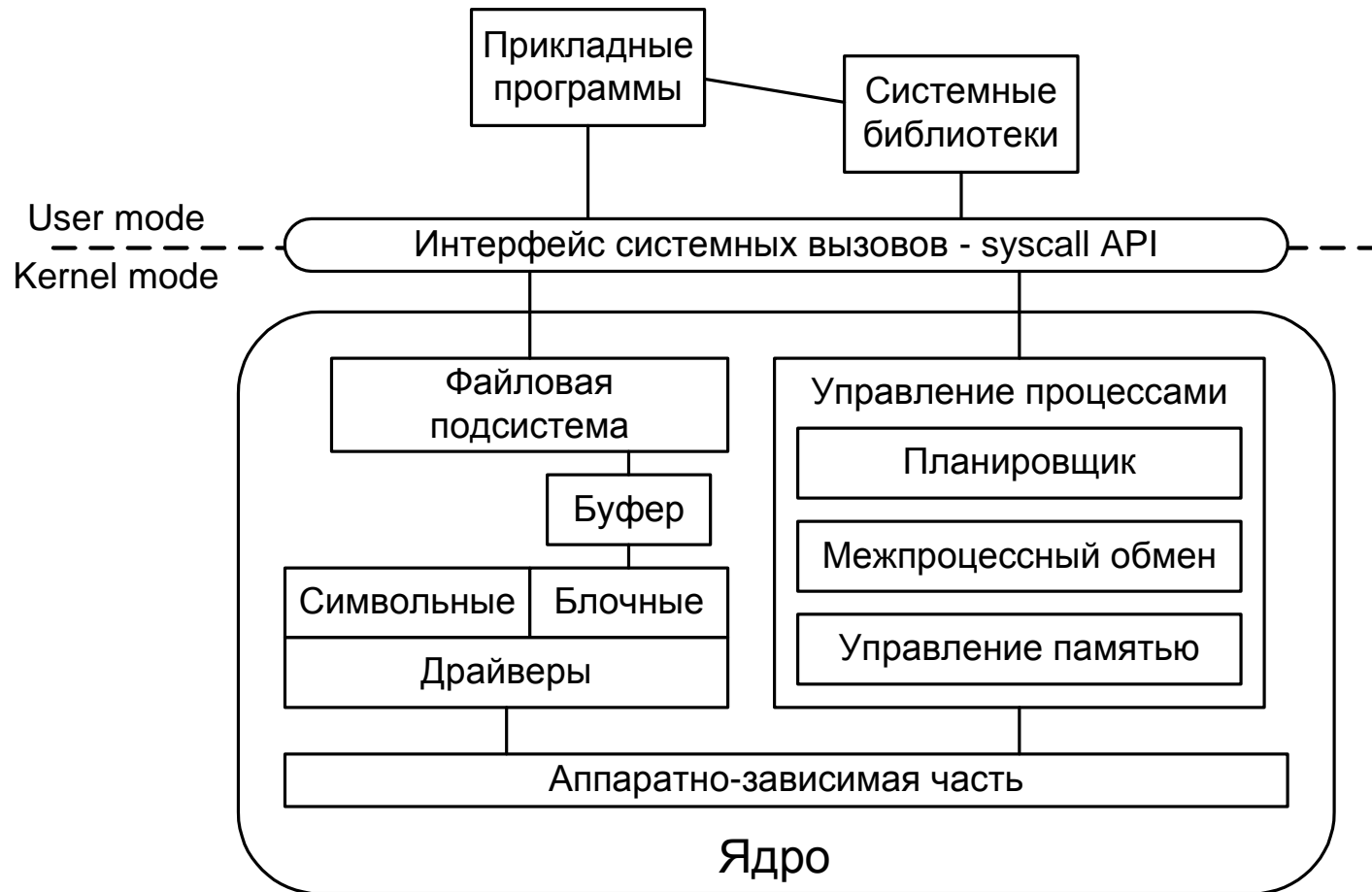
Сектора, представляющие группы процессов, не отражают их реальное количество: набор системных программ ограничен, а разнообразие прикладных потенциально намного больше.

Большинство программ, относимые к системным утилитам, выполняются в **режиме приложений** (***user mode***), то есть наравне с прикладными, но, возможно, от имени привилегированного пользователя.

В частности, «оболочка» (***shell***), несмотря на свое особое положение в системе, является прикладной программой.

«Настоящие» системные процессы порождаются специальным образом из их двоичных файлов под управлением процесса ***init*** и других системных процессов.

## Более развернутое представление архитектуры традиционной Unix-системы:



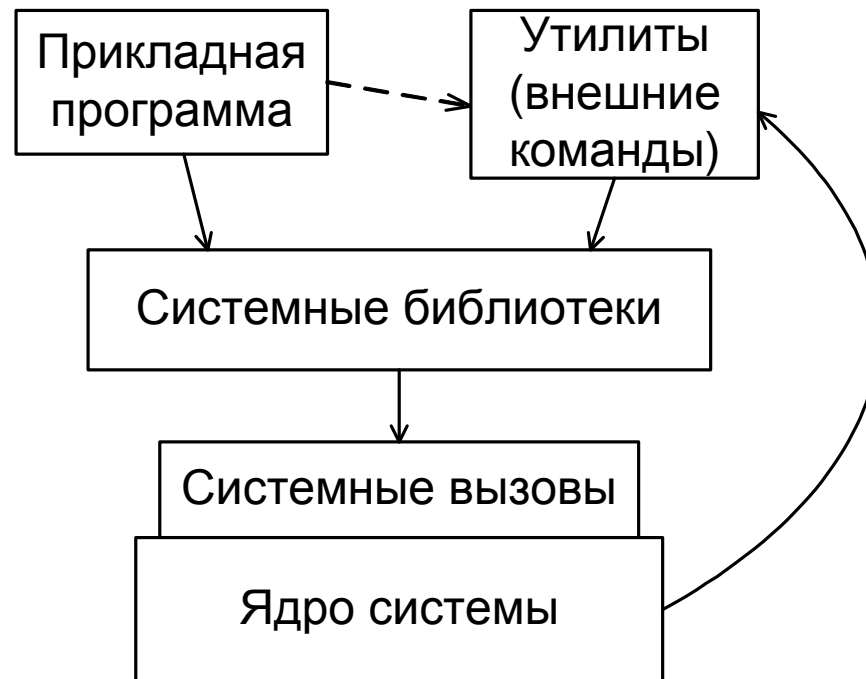
Традиционная архитектура Unix-системы (макроядро)

Существуют также Unix-подобные системы с архитектурой микроядра, например: OSF/1 (микроядро Mach), Chorus, QNX.

С точки зрения программ доступ к системе опосредован вызовами **библиотечных функций**, многие из которых служат «обертками» **системных вызовов** (***syscalls***) или обращаются к ним. Посредством этой цепочки программа может также обращаться к другим самостоятельным программам – «внешним командам» ОС.

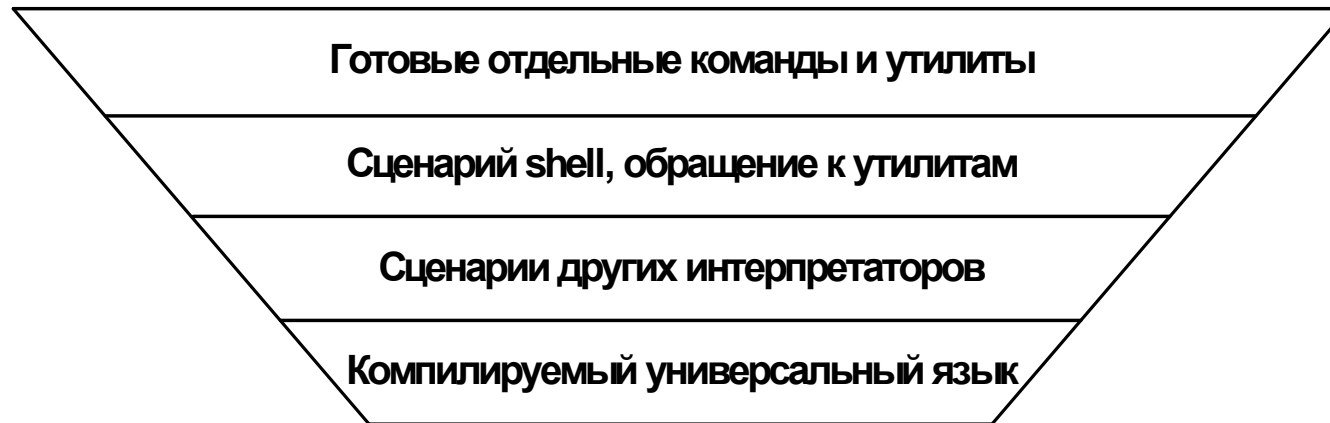
В свою очередь, система предоставляет **пользователю** свои ресурсы в виде **процессов** и **файлов**.

## Операционные системы и среды: Семейство ОС Unix/Linux



Порядок выполнения функций программы

В мире Unix-систем принято следовать следующей иерархии подходов к решению задачи



«Уровни» решения задачи

Пирамида уровней опрокинута – по мере продвижения «вглубь» уровней, часть задач оказывается уже решенной на предыдущих уровнях.

Одна из основных идеологических концепций в Unix – решение сложной задачи достигается взаимодействием нескольких программ, каждая из которых качественно и эффективно решает свою более простую узкую задачу, но не «знает» об общем алгоритме. Цельное конечное решение формируется в результате правильного сочетания и взаимодействия решений элементарных подзадач.

«Передний край» при получении доступа к системе – программа-«оболочка», командный интерпретатор, ***shell***. Он распознает и выполняет команды – внутренние и внешние, в интерактивном или пакетном режиме.

## 1.4 Практическая часть: Сеанс, средства удаленного доступа

**Сеанс** (*session*) – повтор определения: все процессы пользователя, выполняемые начиная с момента входа в систему (login) и до выхода из нее (logout), связанные с одной консолью.

**login** – обеспечивается соответствующими демонами, связанными с консолями. Пользователь не может влиять на этот процесс.

**logout** – выполняется по инициативе пользователя, обращением к системному вызову или команде.

Консоль для удаленного доступа – традиционно называют «**терминалом**». Программы, обеспечивающие виртуальные консоли – также традиционно «терминалы».



Существуют различные протоколы удаленного доступа с соответствующими программными реализациями.

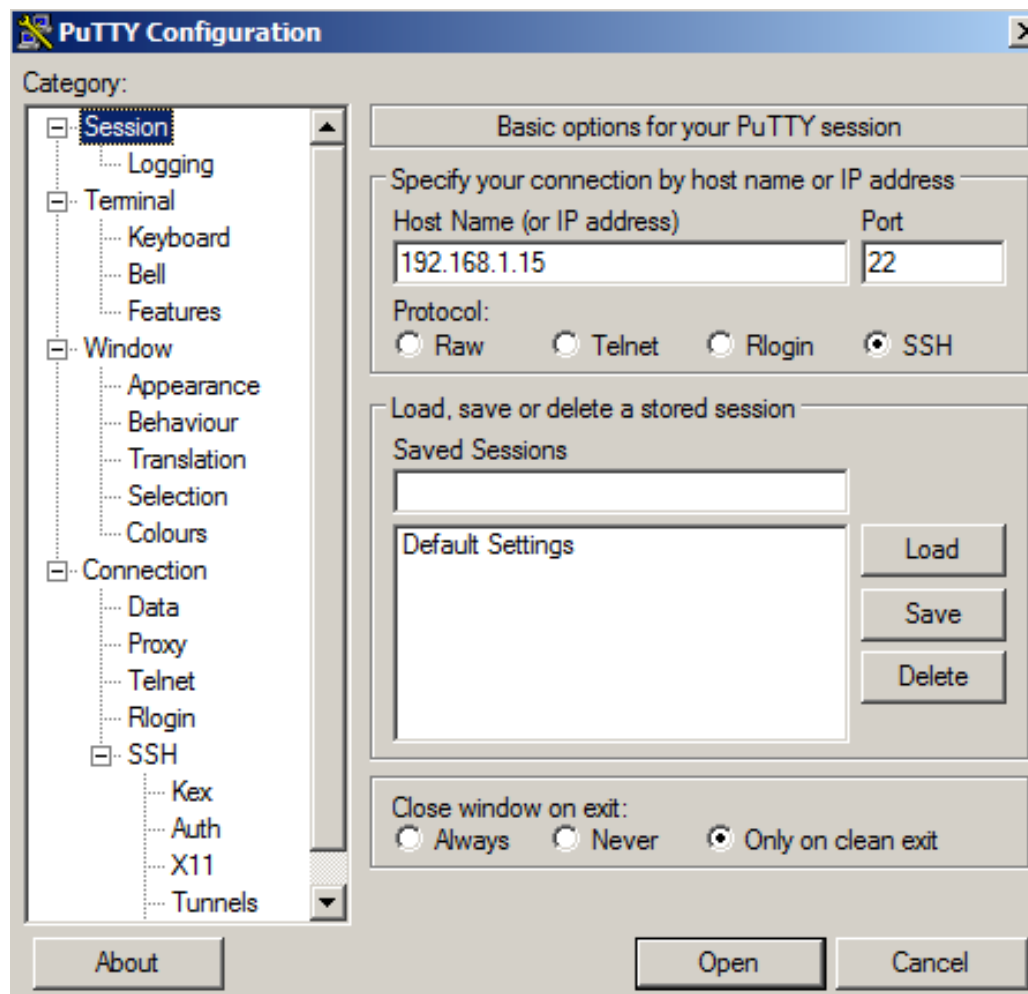
В стеке сетевых протоколов они занимают верхний прикладной уровень (application layer).

***telnet*** – исторически более ранний терминал: сетевой протокол, реализующая его служба (сервер) и программа-клиент. До сих пор входит в стандартный набор «сетевых» программ системы, но рассматривается как небезопасный: протокол дает широкие возможности управления системой, но при этом не предусматривает шифрования трафика между клиентом и сервером, что позволяет «подделать» команды пользователя, перехватить и использовать его аутентификационные данные и т.д. Тем не менее, программа-терминал telnet, реализующая этот протокол, обычно присутствует в системах и может использоваться в качестве «универсального» ТСП-клиента.

**SSH** (**S**ecure **S**hell) – сетевой протокол безопасного удаленного входа и управления системой, предусматривает шифрование трафика и ряд других возможностей, в конечном итоге позволяет удаленно выполнять команды, подобно *telnet*. Программа-клиент `ssh` – утилита командной строки, подобная *telnet*. Клиент `PuTTY` – оконное приложение.

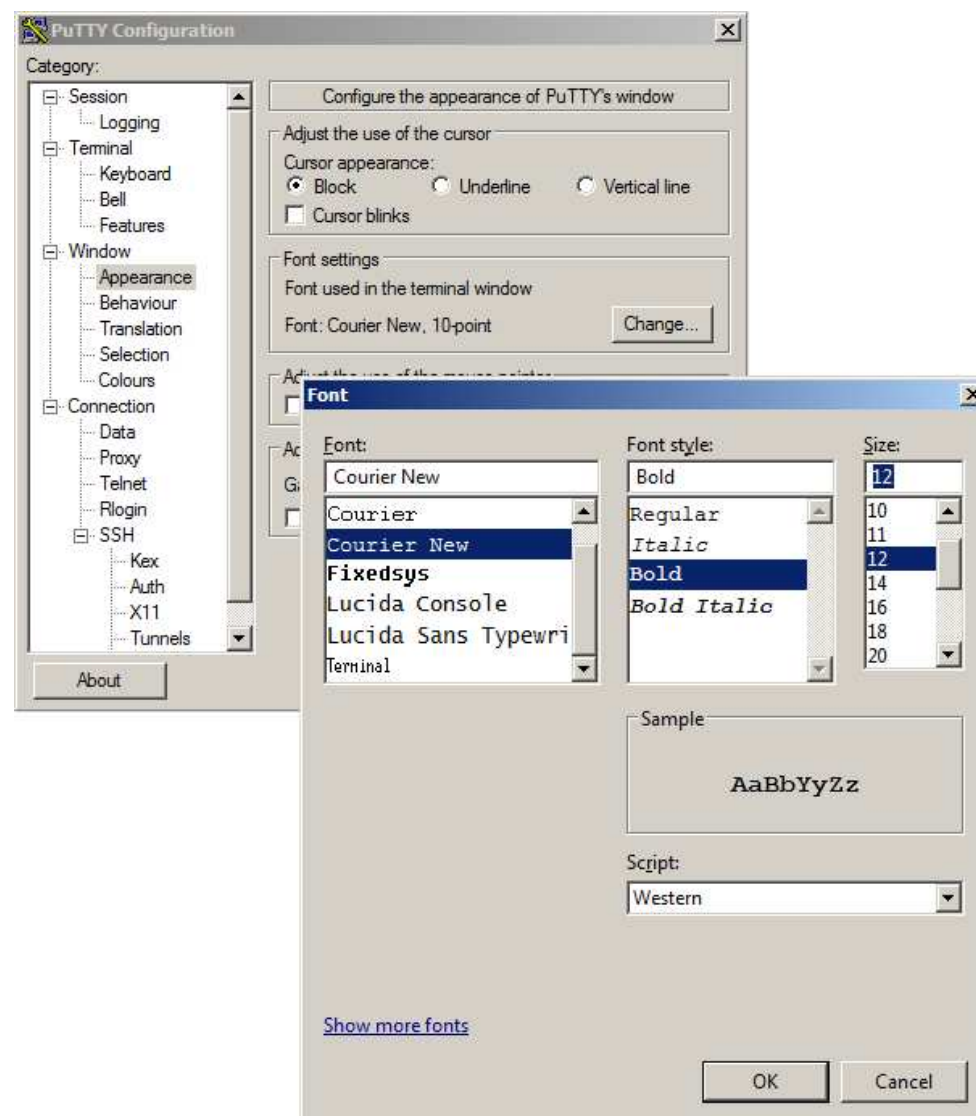
**SCP** (**S**ecure **C**o**P**y) – протокол безопасного копирования файлов удаленной системы, фактически доступа к ее файловой системе, использующий в качестве транспорта SSH. Клиенты: `scp` (командная строка), `WinSCP` (оконное приложение) и др. Совместно с `PuTTY` они позволяют организовать полноценное взаимодействие с удаленной системой.

## Операционные системы и среды: Семейство ОС Unix/Linux



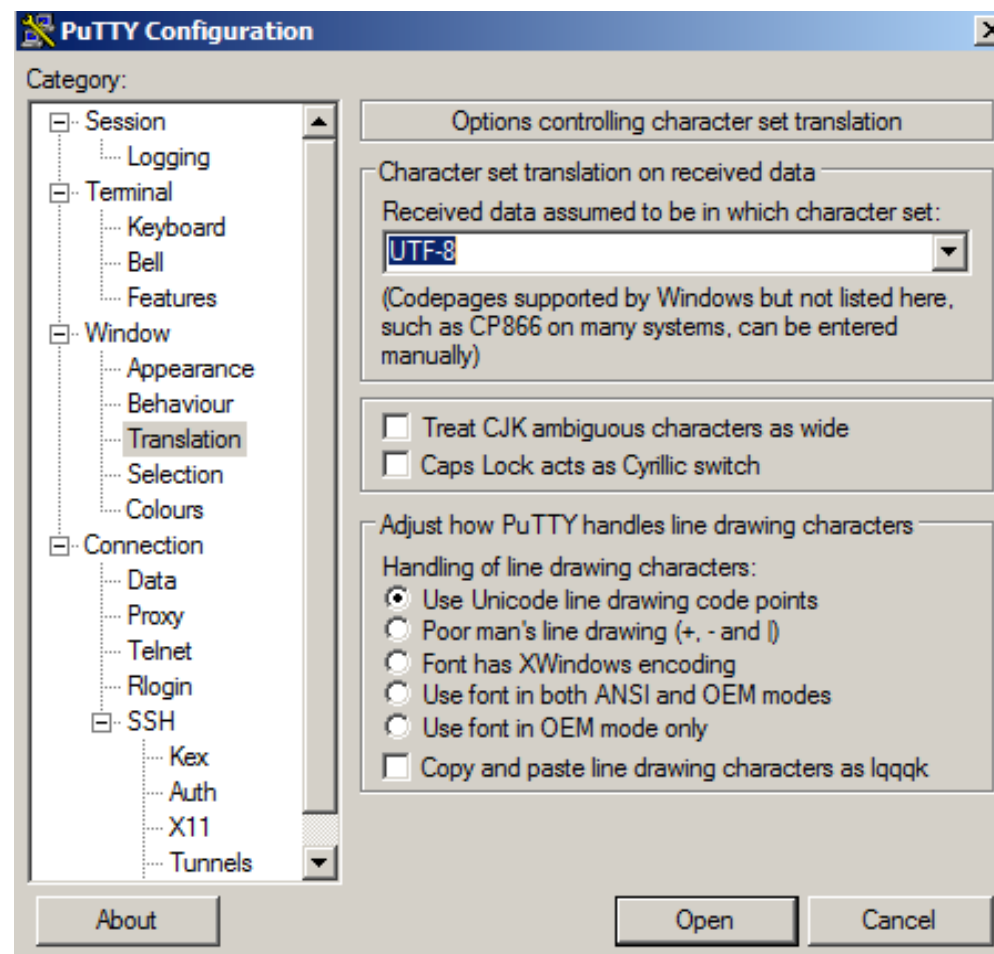
PuTTY – окно параметров соединения

## Операционные системы и среды: Семейство ОС Unix/Linux



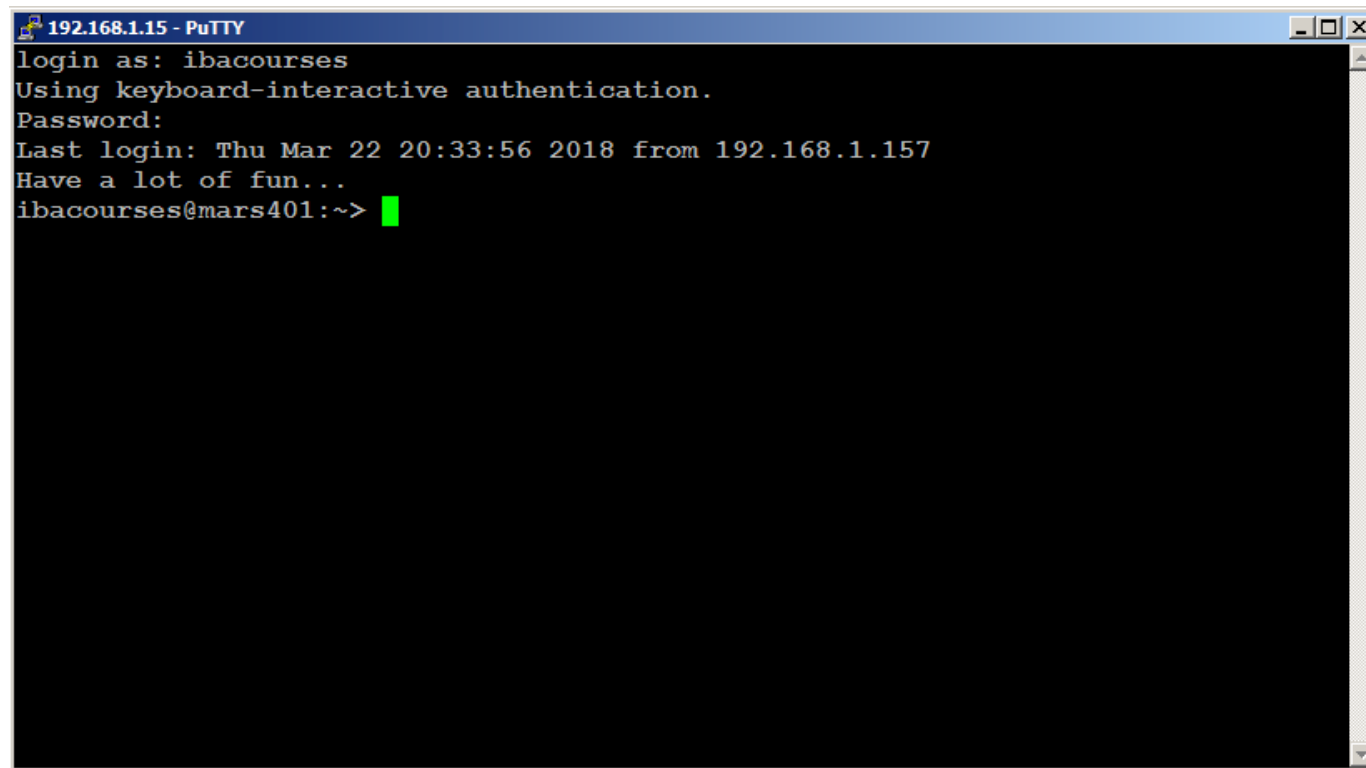
PuTTY – окно параметров шрифта (если требуется)

## Операционные системы и среды: Семейство ОС Unix/Linux



PuTTY – окно параметров кодировок текста

## Операционные системы и среды: Семейство ОС Unix/Linux



```
192.168.1.15 - PuTTY
login as: ibacourses
Using keyboard-interactive authentication.
Password:
Last login: Thu Mar 22 20:33:56 2018 from 192.168.1.157
Have a lot of fun...
ibacourses@mars401:~>
```

PuTTY – рабочее окно сеанса

Параллельно может существовать множество сеансов, в т.ч. и одного и того же пользователя – каждый в своем «терминале».